Making Your Application Unicode Savvy

Session 200

Making Your Application Unicode Savvy

Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations

Making Your Application Unicode Savvy

Deborah Goldsmith
International and Text Group

Agenda

- Overview of Unicode and Mac OS X support
- How to support Unicode in your application

Unicode and Mac OS X

The Past

- WorldScript
  - Each script has its own encoding
  - Some have more than one (Roman, Arabic)
  - Based on standards, but not standard
- Encoding implied by font ID range
- Wrong font ⟶ gibberish
  - ∆ƒø ˆ «√∑Ø±◊¿Œ

Unicode to the Rescue!

- A worldwide standard
  - ISO and Unicode Consortium
  - Covers most writing systems
- Each character has its own unique code point
  - An A is an A is an A
- Unicode 3.2: 95,156 graphic characters

Mac OS X: Unicode Advantages

- One character set for all languages
- More characters for existing languages
- More languages
- No garbled text
- Multiple languages at once
- Cross platform

Unicode Coverage: Alphabetic

- Lucida Grande
  - Extended Roman
  - Cyrillic
  - Vietnamese
  - Greek
- Times, Helvetica, etc.
  - Extended Roman

Unicode Coverage: Japanese

- Hiragino: 6 DTP quality OpenType Type 1 fonts
- Industry-leading character coverage
  - Full JIS X 0213
  - Adobe Japan 1-4
  - Shaken 78 Phototypesetting Kanji
  - NLC shape recommendations
    - (国語審議会表外漢字字体表)

Unicode Coverage: Japanese

- Over 20,000 glyphs vs. ≈7,000 in MacJapanese
- Gaiji problem greatly reduced
    - Data reusable and cross platform
- Can be used in HI, not just documents
- Coming soon: variant glyph access via TSM

Demo

Japanese and Unicode

Unicode Coverage: Jaguar Plans

- Chinese
  - GB 18030 fonts: 32,000+ glyphs
  - CJK Unified Ideographs
  - Ideographic Extension A
  - Yi
  - Partial coverage of Tibetan, Mongolian

Unicode Coverage: Jaguar Plans

- Arabic, Hebrew, Thai, Devanagari, Gurmukhi, Gujarati, Icelandic, Turkish, Greek, Croatian, Romanian, Slovenian, Hawaiian
- All future scripts only via Unicode
  - No WorldScript I or Roman variant scripts
  - No extensions to WorldScript II

Language Support Requirements

- Fonts
- Input method or keyboard
- Collation (comparison) override
  - Default is Unicode order (UTS #10)
- Date/Time/Currency

Adding a Language: Fonts

- Valid, comprehensive Unicode cmap
- Valid ‘post’ table
- Valid ‘name’ table
  - PostScript, Unique, Full, Family, Style, Version
- Valid ‘OS/2’ table
  - ulUnicodeRange and ulCodePageRange
- ‘morx’ table if shaping behavior needed
  - Unicode composition can be synthesized

Adding a Language: Keyboards

- New: drop-in keyboard layouts!
  - /Library/Keyboard Layouts or ~/Library/...
- Unicode keyboard layout defined via XML file
- Equivalent to 'uchr'
- New APIs for manipulating keyboards
  - KLGetKeyboardLayoutxxx, etc.
  - Do not use the Resource Manager!

Demo

XML Keyboard Layout

Unicode in Your Application

Character/Glyph Model

- Critical concept for Unicode (see UTR #17)
- Characters represent information (“spoken”)
- Glyphs are shapes on screen or page (“written”)
- Often 1:1, but not always
    - Arabic, Indic, but even English and Japanese
- Unicode rendering system must map characters to glyphs

The Character/Glyph Model

Unicode String

1	प	092A
2	ू	0942
3	र	0930
4	्	094D
5	त	0924
6	ि	093F

Font Data

Layout

Demo

Typing in Devanagari

Encoding Forms

- Scalar Values (used in HTML/XML)
  - U+0000 through U+10FFFF
- UTF-16 (used in Carbon, Cocoa, Java)
  - One or two 16-bit values per scalar
- UTF-8 (used in BSD)
  - 1–4 bytes per scalar
- UTF-32 (not used in Mac OS X)

Normalization Forms

- Unicode Standard Annex #15
- Fully decomposed (HFS+)
  - e◌́ but not é
- Canonical composed (Internet, Windows)
  - é but not e◌́
  - TEC support planned for Jaguar
- Two more forms (compatibility decompositions)

Text Storage

- Cocoa
  - NSString
- Carbon
  - CFString
  - Raw UniChar (UTF-16) arrays
- Disk (documents, .strings or .plist files)
  - Big-endian UTF-16, or UTF-8

Unicode Pitfalls: Characters

- Clusters
  - é or が
- Surrogates
  - 鰀 (U+9FA5) vs. 𠀋 (U+2000B)
- Multiple “spellings”
  - é vs. e◌́

Problems in Non-Savvy Apps

- Splitting clusters, assuming 1:1:1 code:char:glyph
  - “Long résumé” becomes “Long résume . . ” ·
- Splitting surrogate pairs
  - 𪜈 becomes garbled:  
- Not recognizing canonical equivalents
  - Résumé ≠ Re◌́sume◌́ ≠ Résume◌́ . . · ·

Solutions

- NSString
  - **rangeOfComposedCharacterSequenceAtIndex:**
  - **compare:**
- Unicode Utilities
  - **UCFindTextBreak** (char, cluster, word, line)
  - **UCCompareText**
- ATSUI
  - **kATSULineTruncationTag**

Unicode Pitfalls: Complex Scripts

- Bidirectional and/or cursive
  - Arabic ةيبرعلا → العربية
  - Hebrew עברית
  - Zapfino *Zapfino*
- Rearrangement
  - Devanagari ह ि न् द ी → हिन्दी

Problems in Non-Savvy Apps

- Drawing style runs one at a time: wrong order!

  English العربية العربية English

  English العربية العربية English

- Assuming char index = glyph index
  - Improper hit testing, highlighting, cursor movement

Solutions

- Unicode layout must be for entire paragraph
  - **NSAttributedString**, **NSTypesetter**
  - **ATSUTextLayout** + **ATSUStyle**
- Map between char and coordinate using APIs
  - **ATSUOffsetToPosition**, **ATSUPositionToOffset**
  - **ATSUxxxCursorPosition**
  - **NSLayoutManager**

Supporting Older Encodings

- Which encoding to use?
  - Usually **GetApplicationTextEncoding()**
  - Sometimes **CFStringGetSystemEncoding()**
  - Otherwise, application dependent
- CFString/NSString
- Text Encoding Converter
- Both handle Internet/Windows as well as Mac OS

System Services Summary

- CFString/NSString
  - UTF-16 storage
  - String manipulation
  - Encoding conversion
- NSString
  - Cluster boundaries
  - Locale-sensitive collation

System Services Summary

- Unicode Utilities
  - Boundaries (character, cluster, word, line)
  - Cursor movement (forward/back)
  - Collation (comparison)
  - Locale/Region mapping

System Services Summary

- MLTE
  - Unicode text editing and display
  - Replacement for TextEdit in Carbon
- NSTextView
  - Unicode text editing and display

System Services Summary

- Text Encoding Converter
  - Supports a large number of encodings
  - Multiple forms of Unicode
- Text Services Manager
  - Necessary for Unicode or CJK input
  - Handled for you by MLTE, NSTextView

Summary of Jaguar Plans

- Additional Unicode coverage
  - GB 18030
  - More languages
- Drop-in keyboards
- XML keyboards
- Conversion to precomposed Unicode

Sources of Information

http://www.unicode.org/

- Technical reports, code charts, sample code

- Unicode 3.0 book (ISBN 0-201-61633-5)

http://developer.apple.com/intl

- International Technologies

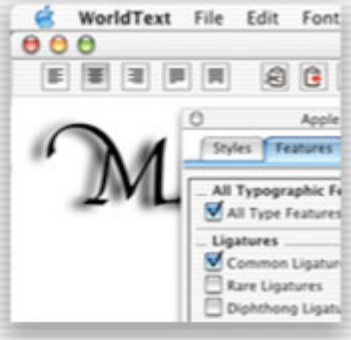
http://developer.apple.com/fonts

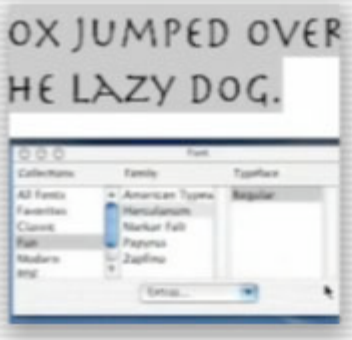
- Font specs and font tools

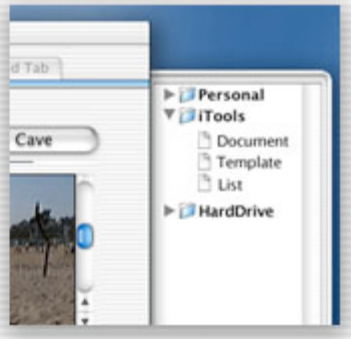
Roadmap

202 Drawing Text With ATSUI: Apple Type Services for Unicode Imaging	Room J **Tue., 3:30pm**
International BoF: Meet the engineers	Room N **Tue., 7:30pm**
208 MLTE: A Unicode Text Engine: The Multilingual Text Engine	Room A2 **Thurs., 9:00am**
FF008 International Feedback Forum: Come tell us what you think!	Room J1 **Thurs., 5:00pm**

Q&A







Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html

Who to Contact

Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html

WWDC2002

WWDC2002

WWDC2002